

**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО УЧЕБНО-ВЫЧИСЛИТЕЛЬНОМУ**

**ПРАКТИКУМУ**

### Студент Завойских Евгения Васильевна

*фамилия, имя, отчество*

### Группа ИУ7-13Б

Тип практики Учебная

Название

предприятия

Студент Завойских Е.В.

*подпись, дата фамилия, и.о.*

### Руководитель практики Кузнецова О.В.

*подпись, дата фамилия, и.о.*

### Оценка

*2020 г.*

**Оглавление**

1. Условие задачи……………………………………………..…………………..3
2. Схема программы……………………………………………………………....4
3. Описание программы…………………………………………………………..7
4. Текст программы…………………………………………………………….....9
5. Результаты работы……………………………………………………………11
6. Заключение……………………………………………………………………13
7. Список литературы…………………………………………………………....13

**Оригинальный текст задачи.**

В известной игре (также называемой "Метаграммой") необходимо из одного слова сделать другое, меняя за один ход по одной букве. Например, сделать из мухи слона можно так: муха - мура - тура - тара - кара - каре - кафе - кафр - каюр - каюк - крюк - урюк - урок - срок - сток - стон - слон.

Напишите программу, которая по данному набору слов, начальному и конечному слову, находит самый короткий способ превратить одно слово в другое.

**Формат входных данных**

Первое число входных данных содержит целое число N, 2≤N≤105. Далее идет N различных строчек, каждая из которых содержит одно словарное слово. Все слова состоят из заглавных латинских букв и имеют равную длину, не превосходящую 10.

Программа должна найти наименьшую цепочку, состоящую из данных слов. Цепочка должна начинаться со слова, идущего в словаре первым и заканчиваться словом, идущим в словаре вторым. Два соседних слова в цепочке должны отличаться ровно в одной букве.

**Формат выходных данных**

Первая строка выходных данных должна содержать количество слов в минимальной искомой цепочке (включая начальное и конечное слово). Далее должны идти все слова цепочки.

Если искомой цепочки не существует, программа должна вывести число 0.

**Пример**

**Вход** **Выход**

|  |  |
| --- | --- |
| 8  BAY  PET  RAY  BET  ONE  TWO  BAT  RAT | 4  BAY  BAT  BET  PET |

# Схема программы

words\_search(word)

**начало**

word = last (последнему слову искомой цепочки)

нет

просматриваем все элементы w списка словарных слов words

да

длина списка now < длины списка result или список result пустой

нет

w нет в текущем списке слов now и w может быть добавлен после word (проверка с помощью функции is\_next)

нет

да

копируем список now в список result

да

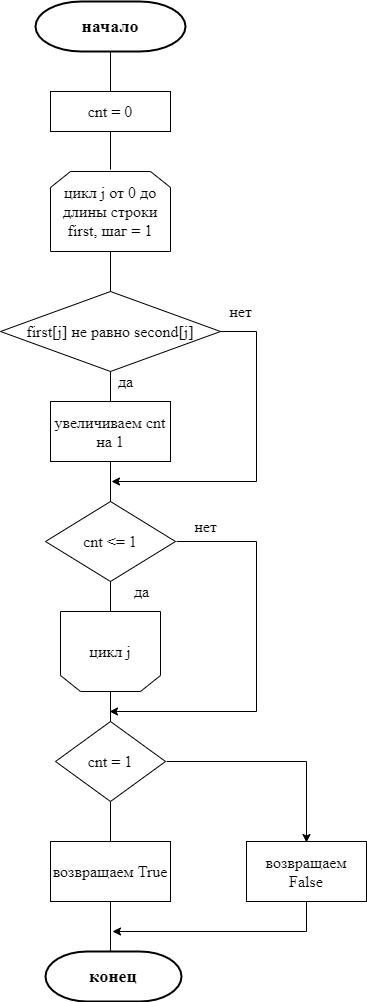
добавление w в конец списка now

words\_search(w)

удаление w из списка now

**конец**

is\_next(first, second)



Основная программа

**начало**

ввод числа словарных слов n и первого слова first

создание пустого списка словарных слов words

ввод словарных слов (кроме первого) в список words

создание last (последнее слово искомой цепочки) - первое в списке words

создание пустого списка result (список для искомой цепочки)

создание списка now (для текущей цепочки) из first

вызов функции words\_search

для first

вывод размера списка result и его содержимого

**конец**

### **Описание программы**

### Считываем число словарных слов и сами слова.

### Чтобы найти наименьшую цепочку слов, начинающуюся с первого словарного слова и заканчивающуюся вторым, мы вызываем функцию words\_search(), единственным параметром которой является строка - первое слово.

Функция перебирает все возможные цепочки слов, начинающиеся с первого словарного слова, добавляя в текущий список слов now[] новое словарное слово после сравнения с последним в списке (переданным в качестве параметра) с помощью функции is\_next(), параметрами которой являются эти два слова. Функция проверяет, чтобы оба слова отличались на одну букву на одной позиции.

Для добавленного слова вызывается функция words\_search().

Если после переданного в функцию слова нельзя добавить новое, выполнение функции для него завершается, мы возвращаемся в место, где она была вызвана, затем это ранее добавленное слово удаляется из списка now[].

Если добавленное слово является последним для искомой цепочки, но список слов now[] не является самым коротким из найденных ранее или единственным, выполнение функции для слова завершается. Мы так же возвращаемся в место, где она была вызвана, затем ранее добавленное слово удаляется из списка now[].

Если добавленное слово является последним для цепочки, и список слов now[] является самым коротким из найденных ранее или единственным, он копируется в итоговый список result[]. Затем выполнение функции для слова завершается. Мы возвращаемся в место, где она была вызвана, затем ранее добавленное слово удаляется из списка now[].

После удаления слова из списка now[], мы продолжаем перебор для других словарных слов.

Когда перебор для всех словарных слов будет выполнен, функция закончит свою работу.

В конце программы происходит печать размера минимальной цепочки и самого искомого списка слов result[] (если нужный список не нашелся, печатается 0).

**Текст программы**

# n - число словарных слов  
# first - первое слово искомой цепочки  
# last - последнее слово искомой цепочки  
# words - список словарных слов (без первого)  
# result - список слов искомой цепочки  
# now - текущий список слов  
# word - последнее слово в текущем списке слов  
# cnt - число различных букв в словах на одинаковых позициях  
  
# Функция, перебирающая все возможные цепочки слов и выбирающая наименьшую подходящую  
def words\_search(word):  
 # Переданное слово - последнее слово искомой цепочки  
 if (word == last):  
 # Текущая последовательность наименьшая из найденных ранее или первая  
 if (len(now) < len(result) or len(result) == 0):  
 result[:] = now[:]  
 else:  
 # Проверяем все словарные слова  
 for w in words:  
 # Такого слова еще нет в текущем списке и оно подходит  
 if (w not in now) and (is\_next(word, w)):  
 now.append(w) # Добавляем слово в текущий список  
 words\_search(w) # Перебор всех возможных цепочек с новым словом в конце  
 now.remove(w) # Удаление слова из текущего списка после всех переборов  
  
# Функция, проверяющая может ли второе переданное слово следовать после первого слова  
def is\_next(first, second):  
 cnt = 0  
 # Перебор всех букв в словах  
 for j in range(len(first)):  
 # Две буквы на одной позиции различны  
 if first[j] != second[j]:  
 cnt += 1  
 # В словах не более одной различной буквы на одинаковых позициях  
 if cnt <= 1:  
 pass

else:

break  
  
 # Второе слово может следовать после первого  
 if cnt == 1:  
 return True  
 # Второе слово не может следовать после первого  
 else:  
 return False  
  
  
n = int(input())  
first = input()  
words = []  
# Считываем n - 1 словарное слово  
for i in range(n - 1):  
 words.append(input())  
last = words[0]  
  
result = []  
# Любая цепочка начинается с первого слова искомой  
now = [first]  
# Перебор всех возможных цепочек, начинающихся с первого слова искомой цепочки, и выбор минимальной  
words\_search(first)  
  
print(len(result)) # Вывод длины искомой цепочки

# Вывод слов искомой цепочки  
for x in result:   
 print(x)

# 

# Результаты работы

**Вход** **Выход**

|  |  |
| --- | --- |
| 8  BAY  PET  RAY  BET  ONE  TWO  BAT  RAT | 4  BAY  BAT  BET  PET |
| 9  FAA FAA SSS DDD FFF GHH JKL ASD FGH | 1 FAA |
| 10 WAY JJK WAK WIK JIK JJK AAA JAY JJY JJK | 4 WAY JAY JJY JJK |
| 5 ZZZZZZZZZZ XZZZZZZZZX ZZZZZZZZZX XZZZZZZZZX ZZZZZZZZXZ | 3 ZZZZZZZZZZ ZZZZZZZZZX XZZZZZZZZX |

# 

# 

|  |  |
| --- | --- |
| 6 ASKI ISKA IIII ISKI ASKA KLKS | 3 ASKI ISKI ISKA |
| 5  AA  BB  AC  CB  ZZ | 0 |

# 

# 

# Заключение

### После работы с программой на поиск минимальной последовательности с определенным условием следования друг за другом слов, я научилась: использовать рекурсию для перебора всех подходящих по условию последовательностей, сравнивать строки между собой, в том числе поэлементно, писать понятный код и грамотно называть переменные.

Я уверена, что эти знания пригодятся мне в дальнейшем обучении.

**Список литературы**

1) Лекции по курсу Борисов С. В.